An Optimization-based Motion Planner for Safe Autonomous Driving

An Optimization-based Motion Planner for Safe Autonomous Driving

Francisco Eiras*, Majd Hawasly*, Stefano V. Albrecht*†, Subramanian Ramamoorthy*†
*FiveAI Ltd, UK, {first.last}@five.ai †School of Informatics, University of Edinburgh, UK

***Abstract*—Guaranteeing safety in motion planning is a crucial bottleneck on the path towards wider adoption of autonomous driving technology. A promising direction is to pose safety requirements as planning constraints in nonlinear optimization problems of motion synthesis. However, many implementations of this approach are hindered by uncertain convergence and local optimality of the solutions, affecting the planner's overall robustness. In this paper, we propose a novel two-stage optimization framework: we first find the solution to a Mixed-Integer Linear Programming (MILP) approximation of the motion synthesis problem, which in turn initializes a second Nonlinear Programming (NLP) formulation. We show that initializing the NLP stage with the MILP solution leads to better convergence, lower costs, and outperforms a state-of-the-art Nonlinear Model Predictive Control baseline in both progress and comfort metrics.**

I. Introduction

The multi-dimensional objective of safe motion planning in autonomous driving is inherently hierarchical in its requirements [6]: the core concern of collision avoidance with road users and obstacles is an *inviolable* hard constraint, while other desired qualities, such as progress towards a destination or passenger comfort, imply softer constraints. In Schwarting et al. [10], motion planning is formulated as a constrained optimization problem in a Nonlinear Model Predictive Control (NMPC) scheme, producing kinematically feasible, safe and smooth trajectories when the MPC stages converge. However, as the authors note, uncertain, locally-optimal convergence is a challenge even when using state-of-the-art solvers [10].

We propose a two-stage optimization method that preserves the prioritization of constraints while mitigating these issues in particular. Specifically, we pose the problem in terms of a first stage modeled as a Mixed-Integer Linear Program (MILP), the output of which initializes a second Nonlinear Programming (NLP) stage. The informed initialization offered by the first stage is an approximate, globally ε-optimal solution to the linearized problem [3, 7] (up to solver tolerances and a user-defined receding horizon). This facilitates the task of the subsequent stage to produce a safe, smooth and kinematically-feasible trajectory. While our framework is similar to [10] in the formulation of the nonlinear problem, we solve two optimization problems at each planning time instead of utilizing an NMPC scheme, initializing the NLP solver with the MILP solution.

II. Two-Stage Motion Planning

At each planning time t_0, we aim to produce a plan of N steps over a horizon of $N\Delta t$ for the *ego* vehicle, where

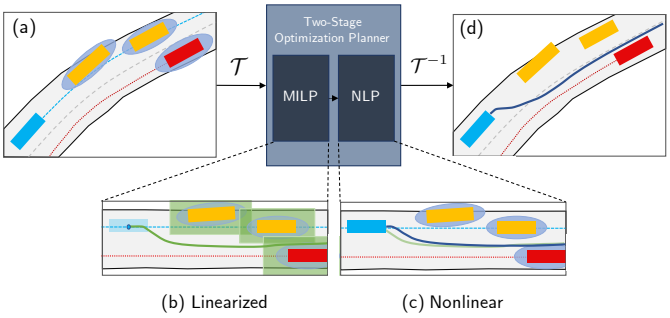
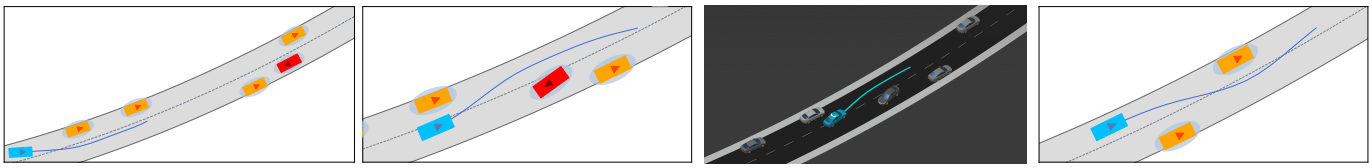


Fig. 1: Two-Stage optimization: (a) from an initial scene, a transform $\mathcal{T}$ yields the planner's input in the reference path representation. The MILP stage (b) solves a linearized version of the problem, which seeds (c) the nonlinear, kinematically-feasible NLP stage. Then, $\mathcal{T}^{-1}$ transforms the output back to a trajectory in the world coordinate frame (d).

Δt is the timestep, such that $t_k = t_0 + k\Delta t$ (step k for shorthand). The input to the planning problem is a scene $\mathbf{s}$ in the world coordinate frame (Fig. 1(a)) which comprises the ego configuration at t_0, road boundaries, timed sequences of posterior distributions of road user poses (parameterized by their expected positions and covariances), and a reference path the ego should follow and progress on, $\mathcal{P}_{\text{ref}}$. The output of the planner is a sequence of ego states $\mathbf{X}_{1:N}$ that attempts to track and progress along $\mathcal{P}_{\text{ref}}$ (Fig. 1(d)). To simplify the planning problem, we project the world coordinate frame using an invertible transform $\mathcal{T}$ to a $\mathcal{P}_{\text{ref}}$-based frame of reference: $\mathcal{T}(\mathbf{s}) = (\mathbf{x}_0, \mathcal{B}, \mathcal{S}^{1:n}_{1:N})$ where $\mathbf{x}_0 \in \mathcal{X}$ is the *nominal* initial ego state, $\mathcal{B} \subset \mathbb{R}^2$ is the driveable surface where it is safe to drive based on the topology of the layout, and $\mathcal{S}^{1:n}_{1:N} \subset \mathbb{R}^{2\times N}$ are unions of elliptical areas that encompass the n road users, $\mathcal{S}^{1:n}_k$, for timesteps $k \in \{1, ..., N\}$. Fig. 1(c) shows the initial timestep after the transform. Then, for a plan $\mathbf{x}_{1:N} \in \mathcal{X}^N$ in the nominal $\mathcal{P}_{\text{ref}}$ coordinate frame (the dark blue trajectory in Fig. 1(c)), we have $\mathcal{T}^{-1}(\mathbf{x}_{1:N}) = \mathbf{X}_{1:N}$, a trajectory in the world coordinate frame (Fig. 1(d)).

To define the planning problem, we consider a discrete dynamic system $\mathbf{x}_{k+1} = f_{\Delta t}(\mathbf{x}_k, \mathbf{u}_k)$, where $\mathbf{x}_k$ is ego state and $\mathbf{u}_k \in \mathcal{U}$ is control (acceleration and steering) applied to the ego at time k. We denote the area the ego occupies at k with $\mathcal{E}(\mathbf{x}_k) \subset \mathbb{R}^2$, and define the cost $J : \mathcal{X}^N \times \mathcal{U}^N \to \mathbb{R}$ as a linear combination of quadratic terms of comfort (reduced acceleration and jerk) and progress (longitudinal and lateral tracking of the reference path, as well as speed). The motion



(a) $t = 0.0s$ (b) $t = 3.2s$ (c) Simulator view ($t = 3.6s$) (d) $t = 5.2s$

Fig. 2: Residential driving example (a), (b) and (d) showing the planner view with ego (blue), static obstacles (orange), oncoming vehicle (red) and ego's plan (dark blue) at different times t; (c) shows our simulator rendering of the situation at $t = 3.6s$. A video of this and other situations are available at https://sites.google.com/view/safe-planning/.

planning problem is then defined as:

$$\begin{aligned} \underset{\mathbf{x}_{1:N}, \mathbf{u}_{0:N-1}}{\text{argmin}} \quad & J(\mathbf{x}_{1:N}, \mathbf{u}_{0:N-1}) \\ \text{s.t.} \quad & \mathbf{x}_{k+1} = f_{\Delta t}(\mathbf{x}_k, \mathbf{u}_k) \\ & \mathcal{E}(\mathbf{x}_k) \cap \left(\left[\mathbb{R}^2 \setminus \mathcal{B} \right] \cup \mathcal{S}_k^{1:n} \right) = \emptyset, \, \forall k \end{aligned} \tag{1}$$

For the vehicle dynamics we consider a kinematic bycicle model, and we approximate the ego's area $\mathcal{E}(\mathbf{x}_k)$ by its corners, so that the intersection with the driveable surface - delimited by its borders which we define as C^2 functions - and road user ellipses can be computed in closed form (similar to [10]).

Eq. (1) is a highly-nonlinear, non-convex optimization problem, and attempting to solve it without a warm start can lead to uncertain convergence and locally-optimal solutions [10, 8, 2, 3]. To mitigate these issues, we first solve a MILP version of the problem by considering a linear dynamical system $\overline{\mathbf{x}}_k = F_{\Delta t}(\overline{\mathbf{x}}_k, \overline{\mathbf{u}}_k)$ under a nonholonomic vehicle model and mixed-integer collision avoidance constraints. In this linear problem **1)** borders are approximated by piecewise-linear functions, and **2)** ellipses of road users are encompassed by axis-aligned boxes (Fig. 1(b)), both of which can be enforced using the *big-M* encoding [13, 9]. Similarly, instead of the quadratic terms, the cost function is approximated by the $|\cdot|$ operator, which can also be implemented in the *big-M* encoding. While MILP algorithms have been shown to guarantee global ε-optimality [3, 7], in practice modern solvers may fail to do so due to rounding errors and built-in tolerances [7]. Furthermore, due to the complexity of this MILP problem, it must be solved in a receding horizon of $K < N$ steps, introducing suboptimality [1, 4]. Nonetheless, a solution close to the global optimum at each receding horizon step acts as a proxy towards a MILP output that is close to the global optimum of the linearized problem, potentially improving the quality of the NLP output, as we show empirically.

III. Experiments

We evaluate our two-stage method by considering 1) alternative heuristic initializations to the NLP stage and 2) an NMPC method similar to [10] in which each MPC stage is initialized by a shifted previous result. We use a dataset of 4000 situations similar to the one in Fig. 2 with a varying number of vehicles, positions and speeds, and base our modeling of other dynamic agents on the Intelligent Driver Model [11].

	% Solved	$\Delta^{\text{NLP}}_{\text{OURS}}$ Cost (%)	$\Delta^{\text{NLP}}_{\text{OURS}}$ Runtime (%)
ZEROS	96.18	11.62	85.36
CT. VEL	64.02	4.11	37.97
CT. ACC	39.98	-0.65	29.73
CT. DEC	93.82	9.64	21.47
OURS (MILP)	**97.99**	-	-

TABLE I: *Initialization Ablation*: percentage of problems NLP solves when initialized with a heuristic [no initialization; constant velocity; constant acceleration; constant deceleration], compared to our MILP initialization, and average improvement in NLP cost/runtime over heuristic in examples solved by both.

	% Solved	P@8s (m)	v (m/s)	$\lvert\dot{a}\rvert$ (m/s^3)
NMPC	87.79	40.93	5.76	0.50
OURS	**98.32**	**52.07**	**6.74**	**0.44**

TABLE II: *NMPC Comparison*: percentage solved, averages of progress after 8s (higher is better), speed (closer to $8m/s$ target is better), and absolute jerk value (lower is better).

As Tab. I shows, our MILP initialization outperforms alternative heuristic choices in the number of problems solved, and results in lower costs and faster convergence in general. Furthermore, as shown in Tab. II, our two-stage method outperforms an NMPC approach (similar to [10]), attempting to optimize the same cost function, in the number of solved examples and in metrics of progress (distance measured along the reference path), velocity tracking and absolute jerk.

While our initial implementation that uses off-the-shelf solvers (Gurobi [5] for MILP and IPOPT [12] for NLP) is slower than an NMPC implementation that utilizes specialized solvers, around $\sim 87.5\%$ of the examples in our experiments were solved in at most $1s$ for an $8s$ horizon, making our framework suitable for deployment in environments such as residential driving.

IV. Conclusion

We introduced an optimization framework for autonomous driving where a linearized version of the planning problem is first solved to initialize a second nonlinear optimization stage. We show that our MILP initialization leads on average to a higher percentage of solved examples, lower cost, and better solving time for the NLP stage when compared to alternative initializations. Additionally, we show that the two-stage formulation solves more examples than an NMPC baseline, outperforming it in terms of progress and comfort metrics.

REFERENCES

[1] Xiaojun Geng and Yugeng Xi. Suboptimality analysis of receding horizon predictive control with terminal constraints. *IFAC Proceedings Volumes*, 32(2):2984–2988, 1999.

[2] Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[3] Ignacio E Grossmann, VT Voudouris, and Omar Ghattas. Mixed-integer linear programming reformuilations for some nonlinear discrete design optimization problems. *Technical Report, Carnegie Mellon University*, 1991.

[4] Lars Grune and Anders Rantzer. On the infinite horizon performance of receding horizon controllers. *IEEE Transactions on Automatic Control*, 53(9):2100–2111, 2008.

[5] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2019. URL http://www.gurobi.com.

[6] Nancy Leveson. Are you sure your software will not kill anyone? *Communications of the ACM*, 63(2):25–28, 2020.

[7] Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13:271–369, 2004.

[8] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[9] Jérémy Omer and Jean-Loup Farges. Hybridization of nonlinear and mixed-integer linear programming for aircraft separation with trajectory recovery. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1218–1230, 2013.

[10] Wilko Schwarting, Javier Alonso-Mora, Liam Paull, Sertac Karaman, and Daniela Rus. Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model. *IEEE Transactions on Intelligent Transportation Systems*, 19(99), 2017.

[11] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805, 2000.

[12] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[13] H Paul Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.